Table of Contents

WLCG Transfer Monitoring

This page documents the WLCG Transfer Monitoring project.

Motivation

- Currently there is no tool which can provide an overall view of data transfer on the WLCG scope (across LHC experiments, across various technologies used, for example FTS and xrootd, across multiple local FTS instances, etc..)
- Every LHC experiment follows it's own data transfer through the VO-specific monitoring system.
- 3 LHC experiments use FTS for data transfer.
- For obtaining data transfer statistics experiments **parse the monitoring pages of the local FTS instances** and/or generate statistics inside the VO-specific monitoring systems.
- Queries of the local FTS instances produce additional load on the local FTS instances.
- **There is a clear similarity between the tasks performed by all VO-specific transfer monitoring systems**. Operations like aggregation of the FTS transfer statistics is done by every VO separately, though can be done once , centrally and then can be served to all experiments via well defined set of APIs
- In order to organize data transfer in a most efficient way experiments would like to have information about FTS queues, about correlations of data transfer between experiments, some other information which is known to FTS but currently not available to the experiments, for example, latencies related to SRM operations during data tranfers, etc...

Instrumentation of the local FTS instances for reporting of the monitoring information

Instrumentation of the FTS instances for reporting of the data transfer information via MSG will allow to :

- broadcast data transfer information to all interested parties avoiding additional load on the local FTS DBs caused by chaotic parsing of the local FTS monitoring pages from many clients
- harmonize client side code and make it independent from the eventual changes of the FTS monitoring UIs

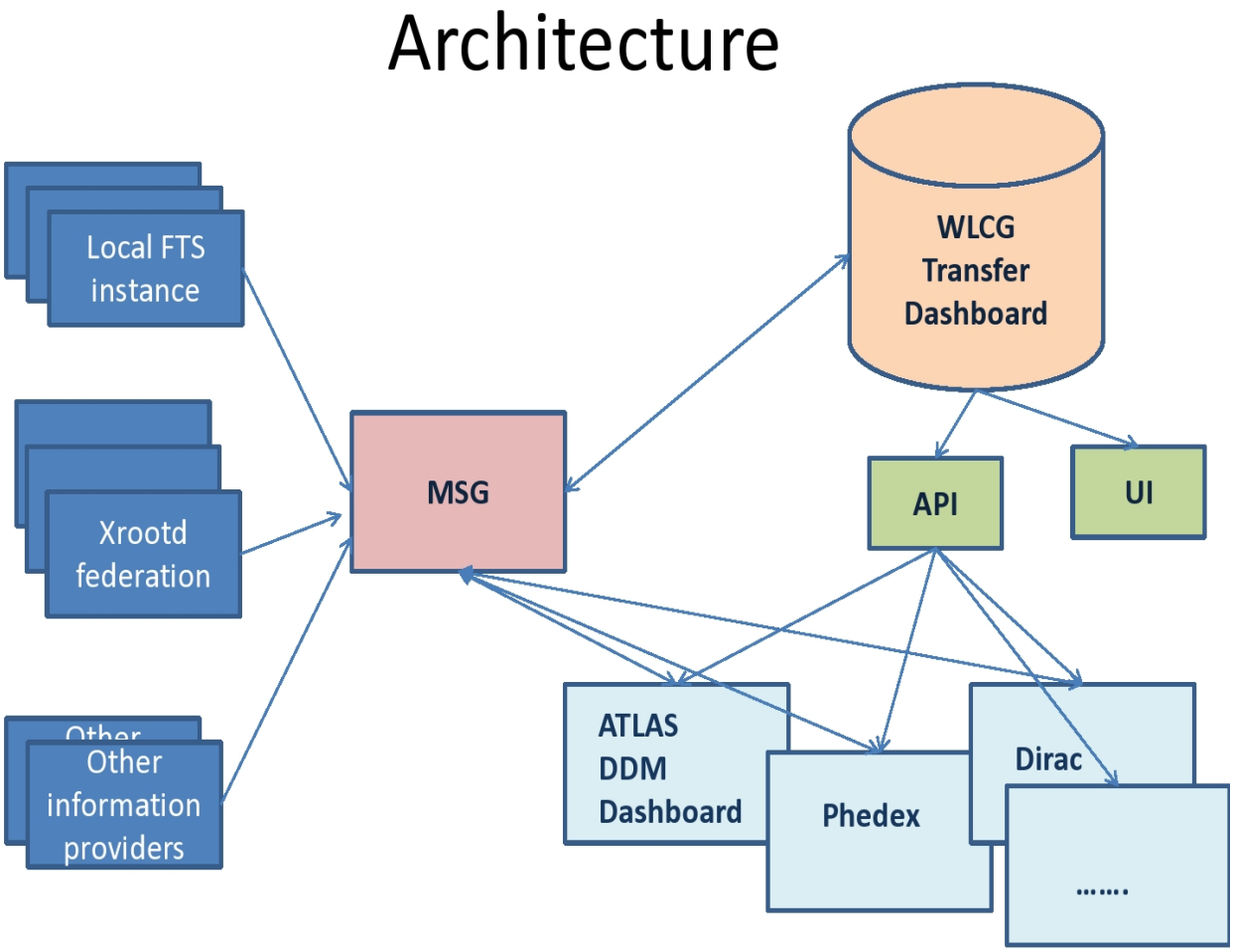
Xrootd federations or any other services dealing with data transfer can be instrumented for the reporting of the data transfer monitoring information via MSG

Data transfer global monitoring system

Information from MSG will be consumed by the central transfer monitoring system which will be responsible for the following tasks:

- perform common tasks as aggregation of the transfer monitoring statistics, generating summaries, etc..
- expose transfer monitoring data to users (via UIs) and other applications (via APIs)

Architecture



Implementation

We have a good chance to make fast progress in the development of the global WLCG transfer monitoring system by re-using a lot of code and experience from the ATLAS DDM Dashboard in terms of schema, statistics aggregation procedures and user interface.

Transfers Dashboard

Infrastructure

- PRODUCTION
  - ♦ hosts: dashb-wlcg-transfers (dashboard63 - dashboard71)
  - ♦ brokers: dashb-mb (gridmsg107 - gridmsg109) fts and gridmsg007 xrootd
  - ♦ fts start queue: /queue/Consumer.dashb.transfer.fts_monitoring_start
  - ♦ fts complete queue: /queue/Consumer.dashb.transfer.fts_monitoring_complete
  - ♦ xrootd atlas queue: /queue/Consumer.dashb_wlcg.xrdpop.fax_popularity
  - ♦ xrootd cms queue: /queue/Consumer.dashb_wlcg.xrdpop.uscms_popularity

- INTEGRATION

-
  - ♦ hosts: dashb-wlcg-transfers-dev (dashboard59)
  - ♦ brokers: dashb-mb (gridmsg107 - gridmsg109) fts and gridmsg007 xrootd

- fts start queue: /queue/Consumer.dashb-int.transfer.fts_monitoring_start
- fts complete queue: /queue/Consumer.dashb-int.transfer.fts_monitoring_complete
- xrootd atlas queue: /queue/Consumer.dashb_dev.xrdpop.fax_popularity
- xrootd cms queue: /queue/Consumer.dashb_dev.xrdpop.uscms_popularity

Links

- WLCG Transfers Dashboard User Interface: http://dashb-wlcg-transfers.cern.ch/ui/
- WLCG Transfers Dashboard Administrator Interface:
  - Collectors Monitoring: http://dashb-wlcg-transfers.cern.ch/ai/insertion_rate.html
  - Stress Test: http://dashb-wlcg-transfers.cern.ch/ai/stresstest.html

Development

See WLCGTransfersDashboard.

Draft of the content of the information which has to be published from FTS to MSG

ATLAS, CMS and LHCb are using FTS for data transfer. Every experiment had developed a monitoring system in order to monitor transfer on the scope of a single experiment. For the moment there is no monitoring tool which provides a single entry point to the transfer monitoring information on the WLCG scope. The first step required in order to develop such a system would be to enable publishing of data transfer monitoring information to MSG.This information can be consumed either by the experiment-specific monitoring systems or by the WLCG global transfer monitoring system. The content of the information which has to be published was discussed among the FTS developers and experiment representatives involved in the development of the data management systems. Below is the first draft describing the content of this information.

There will be two types of messages:

- "Transfer events"
- "Transfer queue"
- Name of the topic/queues has format : transfer*

transfer.start

transfer.complete

transfer.rejected

transfer.queue

For both types of messages every message should contain

- Type of the message A.U. - we will use different queues or topics for different types so we don't need this parameter
- FTS instance identifier
- UTC time stamp of the report

Transfer events can be of two types:

- When transfer starts
- When transfer finishes

"Transfer started": sent when the file moves from "Ready" to "Active" state in the queue.

- Unique ID of the message. This field will include transfer_id plus the FQN of the transfer agent. Also, this field will associate the "transfer started" with "transfer completed" events dublication - we have "Transfer id" and "FQN of the agent initiating the transfer"
- FQN of the agent initiating the transfer
- Transfer id
- Source SURL
- Destination SURL
- Source host
- Destination host new values to be provided

"Transfer complete": sent when the file moves from "Active" to "Done" state. Contains details about the transfer:

- Unique ID of the message. This field will include transfer_id plus the FQN of the transfer agent. Also, this field will associate the "transfer started" with "transfer completed" events dublication - we have "Transfer id" and "FQN of the agent initiating the transfer"
- FQN of the agent initiating the transfer
- Transfer mode(urlcopy, srmcopy). This field will contain info if the transfer is a gridftp transfer(urlcopy) or a direct srmcopy(srmcopy)
- Transfer id
- User DN A.U. - can be moved to "user description"
- User description
- Source SRM version (1.1, 2.0, etc)
- Destination SRM version (1.1, 2.0, etc)
- Source file type (SURM, TURL, URL)
- Destination file type (SURM, TURL, URL)
- VO
- Time stamp transfer_started
- Time stamp transfer_completed
- Transfer duration
- Time stamp checksum_started for source A.U. - don't need this parameter course we have "checksum duration for source"
- Time stamp checksum_started for source A.U. - don't need this parameter course we have "checksum duration for source"
- checksum duration for source
- Time stamp checksum_started for destination A.U. - don't need this parameter course we have "checksum duration for destination"
- Time stamp checksum_started for destination A.U. - don't need this parameter course we have "checksum duration for destination"
- Checksum duration for destination
- Transfer timeout value
- Checksum timeout value
- Total bytes transfered (this will include the info retrieved from the performance markers, no matter if the transfer is successful or not)
- Transfer average throughput (only for gridftp transfers), in kbps
- Final transfer state: OK/Error/Aborted
- Reason of failure, error message, as detailed as it can be
- Failure phase (preparation, transfer, checksum, etc)
- Source or destination failed

There will be two types of messages:

- Number of streams
- Tcp_buffer_size
- Block_size
- File size
- A boolean to indicate if the transfer was interrupted by a user (manual) A.U. - parameter moved to "final transfer state" - Aborted
- Channel used
- Type of the channel (a "dedicated" channel (CERN-CNAF), a "cloud channel" (CERN-T1S) or a "star" channel (CERN-STAR)
- Sites in GOCDB convention linked by the channel A.U. - If we are toking about Source Site Name and Target Site Name we can get this information from "Channel used" parameters. Probably collector will do this work
- Time spent in SRM PREPARATION for the SOURCE
- Time spent in SRM PREPARATION for the DESTINATION
- Time of transfers (physical byte streaming)
- Time spent in SRM FINALIZATION for the SOURCE
- Time spent in SRM FINALIZATION for the DESTINATION
- Time stamp of the event initiation (the time stamp the message was initiated)
- SRM SPACE TOKEN

"Queue status reports" - sampled and sent at regular time intervals (e.g. once per 10 minutes). Also split per VO.

- Channel the report relates to
- type of the channel (a "dedicated" channel (CERN-CNAF), a "cloud channel" (CERN-T1S) or a "star" channel (CERN-STAR).
- sites in GOCDB convention linked by the channel
- Number of files in "Active" transfer state, and Active/Max ratio - on the channel and on each "link" in the channel

where "link" = "(source endpoint, destination endpoint) ordered pair" - e.g. (srm-cms.cern.ch --> cmssrm.fnal.gov)

- Number of files in "Ready" state waiting for transfer - on the channel and on each "link" in the channel

"Transfer started message" draft example (see #FTS_message_structure for current message structure)

```
{
"agent_fqdn" : "fts501.cern.ch", //  FQN of the transfer agent (with Transfer id will be unique i
"transfer_id" : "CERN-GRIDKA__2012-01-16-1746_B4iOjC", // Transfer id
"endpnt" : "https://fts-pilot-service.cern.ch:8443/glite-data-transfer-fts/services/FileTransfer"
"timestamp" : "1326736000000.000000", // UTC time stamp of the report
"src_srm_v" : "2.2.0", // Source SRM version
"dest_srm_v" : "2.2.0", // Destination SRM version
"vo" : "cms", // Virtual organization
"src_url" : "srm://srm-cms.cern.ch/castor/cern.ch/cms/store/PhEDEx_LoadTest07_4/LoadTest07_CERN_2
"dst_url" : "srm://cmssrm-fzk.gridka.de/pnfs/gridka.de/cms/disk-only/store/PhEDEx_LoadTest07/Load
"src_hostname" : "srm-cms.cern.ch", // Source hostname
"dst_hostname" : "cmssrm-fzk.gridka.de", // Destination hostname
"src_site_name" : "CERN-PROD", // Source site name
"dst_site_name" : "FZK-LCG2", // Destination site name
"t_channel" : "CERN-GRIDKA", // Channel used
"srm_space_token_src" : "", // Source SRM SPACE TOKEN
"srm_space_token_dst" : "" // Destiantion SRM SPACE TOKEN
}
```

"Transfer complete message" draft example (see #FTS_message_structure for current message structure)

There will be two types of messages:

```
{
"tr_id" : "CERN-INFN__2012-01-12-1524_TChQ6e",
"endpnt" : "https://fts-pilot-service.cern.ch:8443/glite-data-transfer-fts/services/FileTransfer"
"src_srm_v" : "2.2.0", // Source SRM version
"dest_srm_v" : "2.2.0", // Destination Source SRM version
"vo" : "cms", // Virtual organization
"src_url" : "srm://srm-cms.cern.ch/castor/cern.ch/cms/store/PhEDEx_LoadTest07_4/LoadTest07_CERN_3
"dst_url" : "srm://storm-fe-cms.cr.cnaf.infn.it/cms/store/PhEDEx_LoadTest07/IMPORT/LoadTest07_Deb
"src_hostname" : "srm-cms.cern.ch", // Source hostname
"dst_hostname" : "storm-fe-cms.cr.cnaf.infn.it", // Destination hostname
"src_site_name" : "CERN-PROD", // Source site name
"dst_site_name" : "INFN-T1", // Destination site name
"t_channel" : "CERN-INFN", // Channel used
"channel_type" : "urlcopy" // Transfer mode(urlcopy, srmcopy)
"timestamp_tr_st" : "1326381897000.000000", // Time stamp transfer_started
"timestamp_tr_comp" : "1326381922000.000000", // Time stamp transfer_completed
"tr_timestamp_start" : "1326381886000.000000", // Transfer process start irrespective of failure
"tr_timestamp_complete" : "1326381934000.000000", // Transfer process complete irrespective of fa
"timestamp_chk_src_st" : "1326381886000.000000", // Checksum duration for transfer started
"timestamp_chk_src_ended" : "1326381892000.000000", // Checksum duration for transfer ended
"timestamp_checksum_dest_st" : "1326381934000.000000", // Checksum duration for transfer started
"timestamp_checksum_dest_ended" : "1326381934000.000000", // Checksum duration for transfer ended
"t_timeout" : "3600", // Transfer timeout value
"chk_timeout" : "3600", // Checksum timeout value
"t_error_code" : "", // Error code
"tr_error_scope" : "", // Error scope (SOURCE|DESTINATION)
"t_failure_phase" : "", //Failure phase
"tr_error_category" : "", // Error category
"t_final_transfer_state" : "Ok", // Final transfer state
"tr_bt_transfered" : "2684354560", // Total bytes transfered
"nstreams" : "3", // Number of streams
"buf_size" : "0", // Buffer_size
"tcp_buf_size" : "0", // Tcp_buffer_size
"block_size" : "0", // Block_size
"f_size" : "2684354560", // File size
"time_srm_prep_st" : "1326381886000.000000", // Time spent in SRM PREPARATION for the transfer st
"time_srm_prep_end" : "1326381897000.000000", // Time spent in SRM PREPARATION for the transfer e
"time_srm_fin_st" : "1326381922000.000000", // Time spent in SRM FINALIZATION for the transfer st
"time_srm_fin_end" : "1326381934000.000000", // Time spent in SRM FINALIZATION for the transfer e
"srm_space_token_src" : "", // Source SRM SPACE TOKEN
"srm_space_token_dst" : "", // Destiantion SRM SPACE TOKEN
"t__error_message" : "", // Error description
}
```

"Queue status message" draft example (see #FTS_message_structure for current message structure)

```
{
"fts_id":"https://fts22-t0-export.cern.ch:8443/glite-data-transfer-fts/services/FileTransfer", //
"time":"11:01:36.12", // UTC time stamp of the report
"channels": // array of the channels
[
   {
     "channel_name":"CERN-CERN", // name of the channel
     "channel_type":"", // type of the channel
     "links":  // array of links
       [
        {
          "source_host":"",  // source host of the link
          "dest_host":"", // dest host of the link
          "active":"512", // number of Active transfer
          "ready":"0" // number of Ready transfers
        }
       ]
   },
   {
     "channel_name":"CERN-DESY",
```

There will be two types of messages:

```
      "channel_type":"",
      "links":
        [
         {
           "source_host":"lxbra1910.cern.ch",
           "dest_host":"ennis.desy.de",
           "active":"24",
           "ready":"0"
         },
         {
           "source_host":"lxbra1910.cern.ch",
           "dest_host":"cork.desy.de",
           "active":"47",
           "ready":"0"
         },
        {
           "source_host":"lxbra1910.cern.ch",
           "dest_host":"galway.desy.de",
           "active":"6",
           "ready":"0"
        }
       ]
    }
  ]
}
```


Comments on draft

The following comments on the draft have been received and will be taken into account for the next iteration of the draft. Comments received up to 15/06/2011 are summarised in the section Recommended changes to draft. Any further comments can be added directly in this section or sent to wlcg-transfer-monitor@cernNOSPAMPLEASE.ch

David Tuckett (transcribed from email dated 02/05/2011)

I've reviewed the sample FTS messages given on https://twiki.cern.ch/twiki/bin/view/LCG/WLCGTransferMonitoring

In general, there is more information in the proposed FTS messages than in the current DDM Site Service callbacks, so we should be able to build a monitoring system with at least the same statistics as DDM Dashboard. However, I do have 3 comments.

Firstly, I am concerned as to how we map to the VO specific endpoints from the message fields.

In the started message, we have s_surl, d_surl, s_host and d_host. In the completed message, we have vo, channel and srm_space_token. Somehow these fields must map to vo endpoints. Taking ATLAS as an example, we need to recover something like site_token, e.g. "GRIF-LPNHE_PRODDISK". Perhaps the DQ2 team, who presumably know how this mapping is done in the opposite direction, can comment? [D.T. Added to recommendations as open issue.]

Secondly, what is the rational for choosing which fields are in the started message and which are in the completed message?

For example, vo is in the completed message not the started message. It seems to me that if any of the fields are available at the time of the started message then they should be included there rather than in the completed message. That way, if the completed message is never received we can more easily investigate the issue. [D.T. Added to recommendations.]

Thirdly, I see only a single field for reason of failure. What sort of value do we expect to have in f_reason?

It would be useful to both an error code and the error message (truncated if necessary). [D.T. Added to recommendations.]

Tony Wildish (transcribed from email dated 04/05/2011)

I've taken a look at the transfer monitoring wiki, and have a couple of comments/questions.

1) I suggest putting timestamps in epoch-seconds (or milli/micro seconds) rather than a human-readable time. You probably get epoch time from FTS, and its best to let the client format it if, and only if, they want to. PhEDEx will prefer epoch seconds for the application logic, and will have to convert back to it anyway. [D.T. Added to recommendations.]

2) for transfer states, I believe there are other states that are of interest to us, such as if a job is canceled before it starts. We should have messages for those sorts of conditions. [D.T. The current draft covers the following end states: OK, Error, Aborted. More state transitions can be added if required.]

3) Rather than have a separate message for each transfer-type, I would have a semi-fixed structure, like:

```
transfer_message = {
            type = 'Active',
            other_param1 = 'asdf', // specific to the type
            other_param2 = 1234    // specific to the type
           }
```

so that for all transfer_messages I need only look at the 'type' to know what to do with it. I think this will make it easier to add client-code for new types. I can add a new handler for a new type, and maintain a constant core that simply dispatches different types to their handlers, without having to know what they are. If the type is encoded in the message-name, I can't do that. [D.T. The message type is distinguished by topic/queue name so unwanted messages can easily be ignored. Common attributes between message types should be consistent. e.g. t_id, afqn, time. Added to recommendations.]

E.g, if we decide that 'Pending' or 'Waiting' are also interesting transitions, they can be added, and I can process them, with no impact on my core code.

4) I'm a little unsure of the intent of all the information in the transfer-completed message. There are some redundant values there (transfer started/ended/duration, pick one and drop it!) and others that are not interesting (transfer average throughput). I would avoid calculating quantities that the user may not need and could calculate for themselves if they did. It will reduce your CPU overhead as well as wire-weight of the messages. [D.T. Added to recommendations.]

You could also drop the 'channel type', since the user should be able to look this up once and cache it, it's not needed with every completion message. Presumably such information can be obtained from the API? [D.T. Added to recommendations as open issue.]

5) the 'Transfer id' of a message, does this identify the FTS transfer job, or the individual file within that job? [D.T. Added to recommendations as open issue.]

Recommended changes to draft 15/06/2011

This section contains a summary of the recommended changes to the draft messages based on feedback on the draft messages.

General recommendations

1. Replace all string timestamps by UTC seconds since Unix epoch.
   e.g. '2011-06-15T10:05:53.934122' is replaced by 1308132353.934122
   [JSON has no standard date format and it is easy and efficient to work with UTC epoch seconds. As clarification, see Python example below.]

```
Python 2.6.5 (r265:79063, Apr 16 2010, 13:57:41)
>>> import time
>>> from datetime import datetime
>>> s = time.time()
>>> s # UTC seconds since Unix epoch
1308132353.9341221
>>> d = datetime.utcfromtimestamp(s)
>>> d # Python UTC datetime object
datetime.datetime(2011, 6, 15, 10, 5, 53, 934122)
>>> d.isoformat() # ISO 8601 format string
'2011-06-15T10:05:53.934122'
>>>
```

"Transfer started message" recommendations

1. Move any information that is available at transfer start from "Transfer complete message" to "Transfer started message".
   e.g. "vo", "channel", "srm_space_token", ...

"Transfer complete message" recommendations

1. Add "time" // UTC seconds of the report.
   [To be consistent with other messages.]
2. Add "f_code": 123 // Error code of failure.
   [Assuming such a error code exists, this supplements "f_reason" which is the error message truncated if necessary.]
3. Remove calculated values:
   - "t_duration"
     [Assuming this is just t_completed - t_started.]
   - "throughput"
     [Assuming this is just b_transferred/1024/(t_completed-t_started).]

Open issues

1. How to translate from host, surl, channel, srm_space_token to VO-specific endpoint names such as "GRIF-LPNHE_PRODDISK"? [Julia will check that endpoint identifiers in the messages are sufficient to recover VO-specific names via BDII]
2. Can "t_channel" in "Transfer complete message" be derived from "channel" or an API? [We will leave this in the message even if it is redundant because it simplifies the client code.]
3. Does "t_id" identify an FTS job or an individual file within the FTS job? ["t_id" identifies the FTS job. Michail will investigate how to identify the individual file which the message is about.]

Info about dedicated message broker

Brokers

hostname	state	ActiveMQ version	DNS alias	Monitoring links	Aminstrative UI

gridmsg107	production	5.5.1-fuse-01-06	dashb-mb.cern.ch		https://dashb-mb.cern.ch/admin/index.jsp

gridmsg108	production	5.5.1-fuse-01-06	dashb-mb.cern.ch	https://dashb-mb.cern.ch/admin/index.jsp

Ports

- STOMP+SSL: 6162 (consumer)
- STOMP :6163 (producer)
- OpenWire+SSL: 6167

Use-case details can be found here

Topics and queues on the test broker related to the Global WLCG transfer

use-case	topic	queues
fts transfers	transfer.fts_monitoring_complete	transfer.fts_monitoring_complete
	transfer.fts_monitoring_start	transfer.fts_monitoring_complete
	transfer.fts_monitoring_queue_status	transfer.fts_monitoring_status
		transfer.fts_monitoring_rejected

Topics and queues on the production broker related to the Global WLCG transfer

use-case	topic	queues
fts transfers	transfer.fts_monitoring_complete	Consumer.dashb.transfer.fts_monitoring_complete
	transfer.fts_monitoring_start	Consumer.dashb.transfer.fts_monitoring_start
	transfer.fts_monitoring_queue_status	?
		transfer.fts_monitoring_rejected

Testing queues.

In FTS, every messages arrive in a topic. Then messages are copy in virtual queue and then consumed by the application. This way allow to be sure we don't loose any data even if the collector is not running while messages arrive in the topic. The virual queue is automatically created when consuming from it the first time and stay active FOREVER. This can easily kill the broker if many virtual queues are setup and their messages never consumed.

For devellopement and testing purpose , there is two recommandations to consume. If loosing messages is acceptable, consume directly from the topic. Create a virtual queue and add an entry in the following table otherwise.

Active queues.

broker (or alias)	queue	owner	client	purpose
dashb-mb	Consumer.dashb.transfer.fts_monitoring_start	abeche	dashboard63 (71)	WLCG Transfers Dashboard Production
dashb-mb	Consumer.dashb.transfer.fts_monitoring_complete	abeche		

			dashboard63 (71)	WLCG Transfers Dashboard Production

--+++ Alarms

Alarms should be sent to dashb-mb-alarms@cernNOSPAMPLEASE.ch, curent rule is more than 5000 messages are stuck in the queues

FTS message structure

Sample messages to demonstrate the structure/content:

fts_monitoring_start (extracted 2011-11-18T10:31:03 UTC):


```
{
    "agent_fqdn":"fts501.cern.ch",
    "transfer_id":"RAL-CERN__2011-11-18-1031_OlJgnE",
    "endpnt":"https://fts-pilot-service.cern.ch:8443/glite-data-transfer-fts/services/FileTransfe
    "timestamp":"1321612263000.000000",
    "src_srm_v":"2.2.0",
    "dest_srm_v":"2.2.0",
    "vo":"cms",
    "src_url":"srm://srm-cms.gridpp.rl.ac.uk/castor/ads.rl.ac.uk/prod/cms/store/LoadTest07/LoadTe
    "dst_url":"srm://srm-cms.cern.ch/castor/cern.ch/cms/store/PhEDEx_LoadTest07/LoadTest07_Debug_
    "src_hostname":"srm-cms.gridpp.rl.ac.uk",
    "dst_hostname":"srm-cms.cern.ch",
    "src_site_name":"RAL-LCG2",
    "dst_site_name":"CERN-PROD",
    "t_channel":"RAL-CERN",
    "srm_space_token_src":"",
    "srm_space_token_dst":"CMS_DEFAULT"
}
```


fts_monitoring_complete (extracted 2011-11-18T10:31:17 UTC):


```
{
    "tr_id":"RAL-CERN__2011-11-18-1031_14JIbi",
    "endpnt":"https://fts-pilot-service.cern.ch:8443/glite-data-transfer-fts/services/FileTransfe
    "src_srm_v":"2.2.0",
    "dest_srm_v":"2.2.0",
    "vo":"cms",
    "src_url":"srm://srm-cms.gridpp.rl.ac.uk/castor/ads.rl.ac.uk/prod/cms/store/LoadTest07/LoadTe
    "dst_url":"srm://srm-cms.cern.ch/castor/cern.ch/cms/store/PhEDEx_LoadTest07/LoadTest07_Debug_
    "src_hostname":"srm-cms.gridpp.rl.ac.uk",
    "dst_hostname":"srm-cms.cern.ch",
    "src_site_name":"RAL-LCG2",
    "dst_site_name":"CERN-PROD",
    "t_channel":"RAL-CERN",
    "timestamp_tr_st":"1321612277000.000000",
    "timestamp_tr_comp":"1321612307000.000000",
    "timestamp_chk_src_st":"1321612264000.000000",
    "timestamp_chk_src_ended":"1321612271000.000000",
    "timestamp_checksum_dest_st":"1321612309000.000000",
    "timestamp_checksum_dest_ended":"1321612309000.000000",
    "t_timeout":"1800",
    "chk_timeout":"1800",
    "t_error_code":"",
    "tr_error_scope":"",
    "t_failure_phase":"",
    "tr_error_category":"",
```

```
    "t_final_transfer_state":"Ok",
    "tr_bt_transfered":"2684354560",
    "nstreams":"5",
    "buf_size":"0",
    "tcp_buf_size":"0",
    "block_size":"0",
    "f_size":"2684354560",
    "time_srm_prep_st":"1321612263000.000000",
    "time_srm_prep_end":"1321612276000.000000",
    "time_srm_fin_st":"1321612307000.000000",
    "time_srm_fin_end":"1321612309000.000000",
    "srm_space_token_src":"",
    "srm_space_token_dst":"cms:CMS_DEFAULT",
    "t__error_message":""
}
```


fts_monitoring_queue_status (extracted 2011-11-24T13:45:12 UTC):


```
{
    "fts_id":"https://vtb-generic-32.cern.ch:8443/glite-data-transfer-fts/services/FileTransfer",
    "time":"1322142312000.000000",
    "vo":{
        "voname":"dteam",
        "channel":{
            "channel_name":"CERN-CERN",
            "channel_type":"",
            "links":[
                {
                    "source_host":"lxbra1910.cern.ch",
                    "dest_host":"lxbra2502.cern.ch",
                    "active":"0",
                    "ratio":"0.000000%",
                    "ready":"0"
                },
                {
                    "source_host":"lxbra2502.cern.ch",
                    "dest_host":"lxbra2502.cern.ch",
                    "active":"0",
                    "ratio":"0.000000%",
                    "ready":"0"
                },
                {
                    "source_host":"lxbra1910.cern.ch",
                    "dest_host":"lxbra1910.cern.ch",
                    "active":"0",
                    "ratio":"0.000000%",
                    "ready":"0"
                }
            ]
        },
        "channel":{
            "channel_name":"CERN-DESY",
            "channel_type":"",
            "links":[
                {
                    "source_host":"lxbra1910.cern.ch",
                    "dest_host":"ennis.desy.de",
                    "active":"0",
                    "ratio":"0.000000%",
                    "ready":"0"
                },
                {
                    "source_host":"lxbra1910.cern.ch",
                    "dest_host":"dublin.desy.de",
                    "active":"0",
                    "ratio":"0.000000%",
```

```
                "ready":"0"
            },
            {
                "source_host":"lxbra1910.cern.ch",
                "dest_host":"cork.desy.de",
                "active":"0",
                "ratio":"0.000000%",
                "ready":"0"
            },
            {
                "source_host":"lxbra1910.cern.ch",
                "dest_host":"galway.desy.de",
                "active":"0",
                "ratio":"0.000000%",
                "ready":"0"
            }
        ]
    },
    "channel":{
        "channel_name":"DESY-CERN",
        "channel_type":"",
        "links":[
            {
                "source_host":"dublin.desy.de",
                "dest_host":"lxbra1910.cern.ch",
                "active":"0",
                "ratio":"0.000000%",
                "ready":"0"
            },
            {
                "source_host":"cork.desy.de",
                "dest_host":"lxbra1910.cern.ch",
                "active":"0",
                "ratio":"0.000000%",
                "ready":"0"
            },
            {
                "source_host":"ennis.desy.de",
                "dest_host":"lxbra1910.cern.ch",
                "active":"0",
                "ratio":"0.000000%",
                "ready":"0"
            },
            {
                "source_host":"galway.desy.de",
                "dest_host":"lxbra1910.cern.ch",
                "active":"0",
                "ratio":"0.000000%",
                "ready":"0"
            }
        ]
    },
    "channel":{
        "channel_name":"DESY-DESY",
        "channel_type":"",
        "links":[
            {
                "source_host":"galway.desy.de",
                "dest_host":"galway.desy.de",
                "active":"0",
                "ratio":"0.000000%",
                "ready":"0"
            },
            {
                "source_host":"ennis.desy.de",
                "dest_host":"ennis.desy.de",
                "active":"0",
```

```
                    "ratio":"0.000000%",
                    "ready":"0"
                },
                {
                    "source_host":"cork.desy.de",
                    "dest_host":"cork.desy.de",
                    "active":"0",
                    "ratio":"0.000000%",
                    "ready":"0"
                },
                {
                    "source_host":"dublin.desy.de",
                    "dest_host":"dublin.desy.de",
                    "active":"0",
                    "ratio":"0.000000%",
                    "ready":"0"
                }
            ]
        }
    }
}
```


Proposed change to **fts_monitoring_queue_status** message structure:

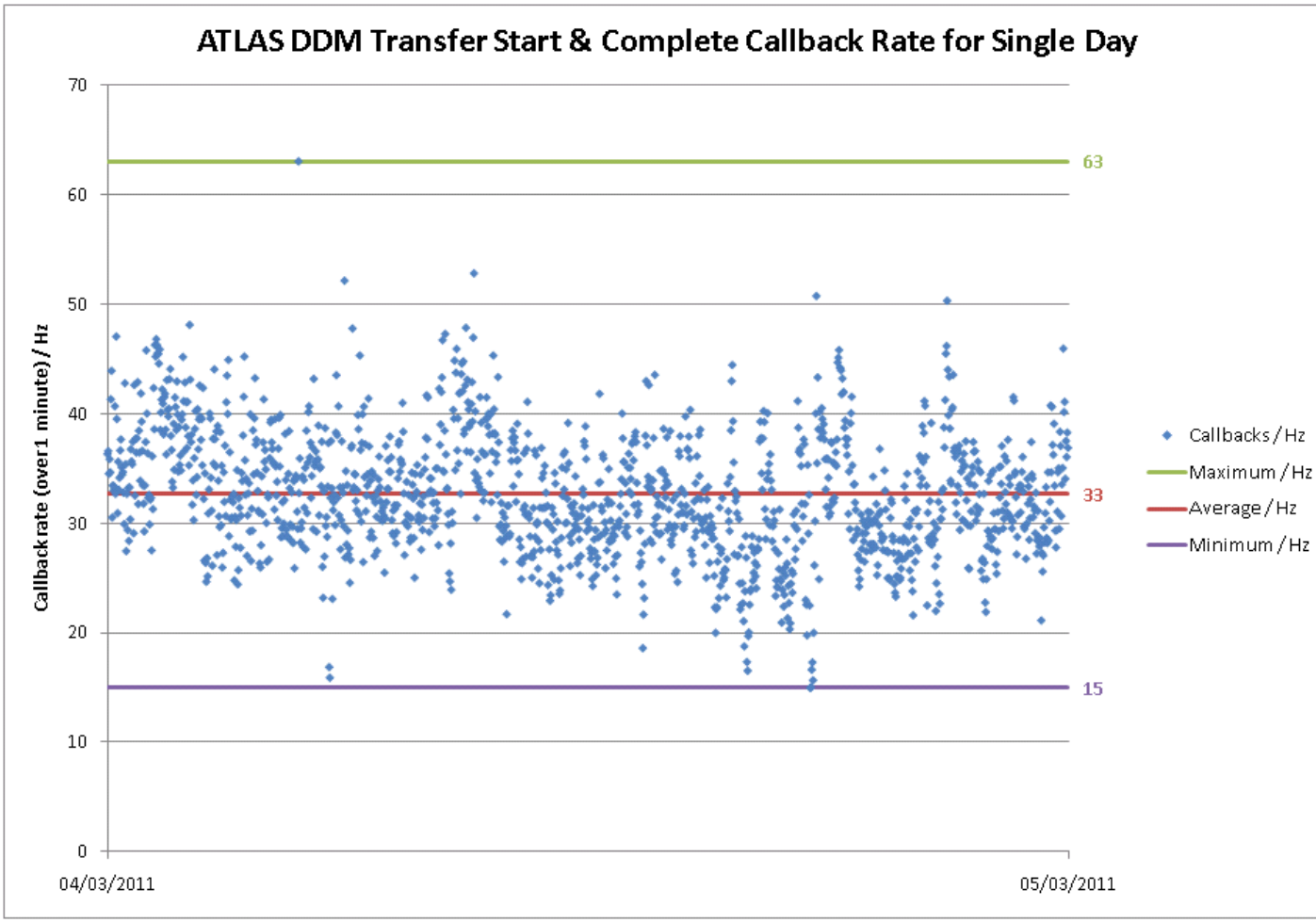

```
{
    "fts_id":"https://vtb-generic-32.cern.ch:8443/glite-data-transfer-fts/services/FileTransfer",
    "time":"1322142312000.000000",
    "vos":[
        {
            "voname":"dteam",
            "channels":[
                {
                    "channel_name":"CERN-CERN",
                    "channel_type":"",
                    "links":[
                        {
                            "source_host":"lxbra1910.cern.ch",
                            "dest_host":"lxbra2502.cern.ch",
                            "active":"0",
                            "ratio":"0.000000%",
                            "ready":"0"
                        },
                        {
                            "source_host":"lxbra2502.cern.ch",
                            "dest_host":"lxbra2502.cern.ch",
                            "active":"0",
                            "ratio":"0.000000%",
                            "ready":"0"
                        },
                        {
                            "source_host":"lxbra1910.cern.ch",
                            "dest_host":"lxbra1910.cern.ch",
                            "active":"0",
                            "ratio":"0.000000%",
                            "ready":"0"
                        }
                    ]
                },
                {
                    "channel_name":"DESY-CERN",
                    "channel_type":"",
                    "links":[
                        {
                            "source_host":"dublin.desy.de",
                            "dest_host":"lxbra1910.cern.ch",
                            "active":"0",
```

```
                        "ratio":"0.000000%",
                        "ready":"0"
                    },
                    {
                        "source_host":"cork.desy.de",
                        "dest_host":"lxbra1910.cern.ch",
                        "active":"0",
                        "ratio":"0.000000%",
                        "ready":"0"
                    },
                    {
                        "source_host":"ennis.desy.de",
                        "dest_host":"lxbra1910.cern.ch",
                        "active":"0",
                        "ratio":"0.000000%",
                        "ready":"0"
                    },
                    {
                        "source_host":"galway.desy.de",
                        "dest_host":"lxbra1910.cern.ch",
                        "active":"0",
                        "ratio":"0.000000%",
                        "ready":"0"
                    }
                ]
            }
        ]
    },
    {
        "voname":"some_other_vo",
        "channels":[
            {
                "channel_name":"CERN-CERN",
                "channel_type":"",
                "links":[
                    {
                        "source_host":"lxbra1910.cern.ch",
                        "dest_host":"lxbra2502.cern.ch",
                        "active":"0",
                        "ratio":"0.000000%",
                        "ready":"0"
                    },
                    {
                        "source_host":"lxbra2502.cern.ch",
                        "dest_host":"lxbra2502.cern.ch",
                        "active":"0",
                        "ratio":"0.000000%",
                        "ready":"0"
                    },
                    {
                        "source_host":"lxbra1910.cern.ch",
                        "dest_host":"lxbra1910.cern.ch",
                        "active":"0",
                        "ratio":"0.000000%",
                        "ready":"0"
                    }
                ]
            }
        ]
    }
]
}
```

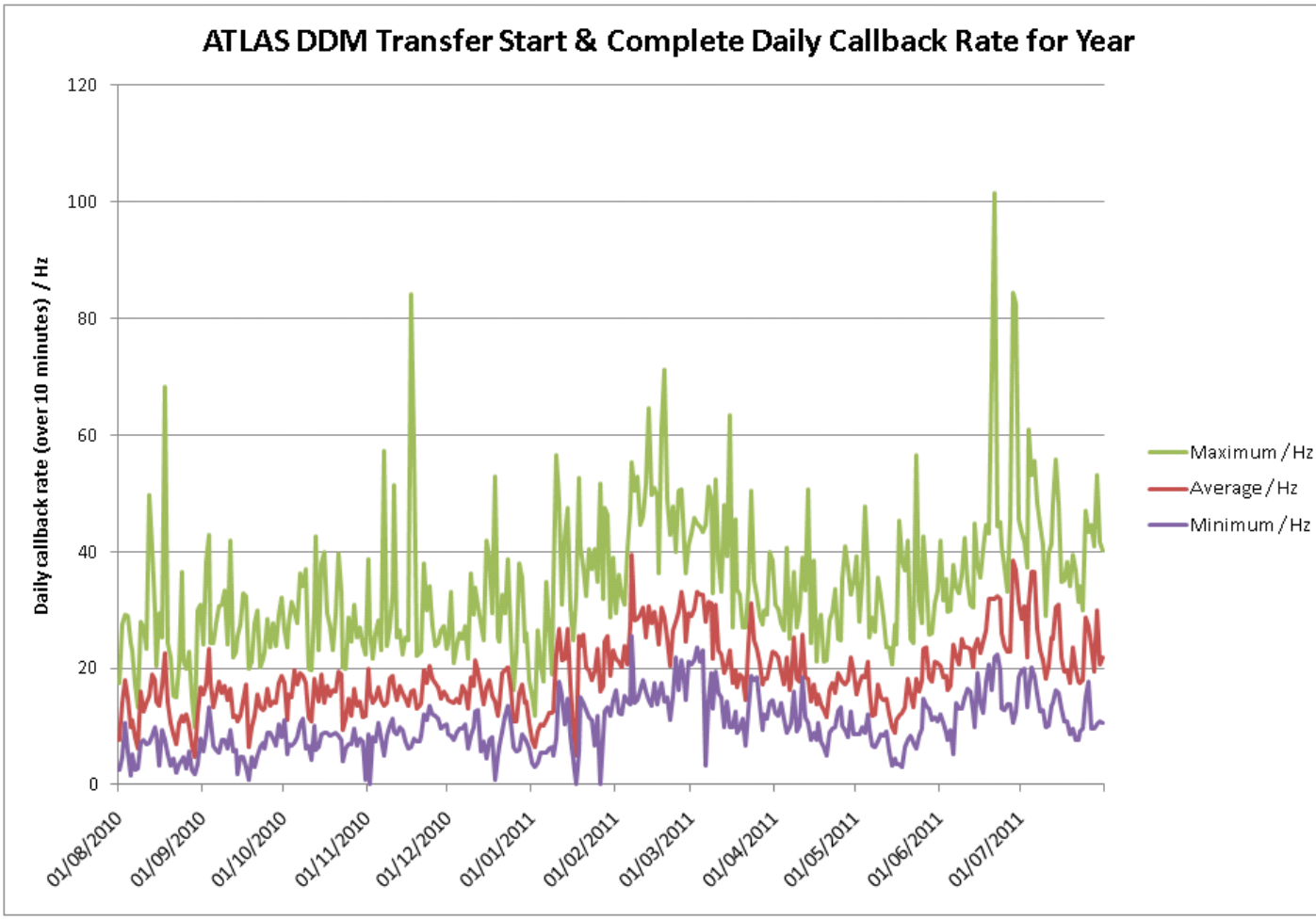
Further Information

ATLAS DDM transfer callback rate

The following shows the rate of transfer callbacks (calculated over 1 minute bins) received by ATLAS DDM Dashboard for 5th March 2011. The chosen day has high load but is not atypical, see next plot.



The following shows the daily average, maximum and minimum rate of transfer callbacks (calculated over 10 minute bins) received by ATLAS DDM Dashboard from August 2010 to August 2011.



Average size of messages from FTS: Start Message: 0.6 kB End Message: 1.6 kB

XROOTD monitoring

All XROOTD monitoring details can be accessed through the following twiki page:

XrootdMonitoring

62526

Apollo testing

All Apollo testing step can be accessed through the following twiki page:

ApolloTesting

62526

This topic: LCG > WLCGTransferMonitoring
Topic revision: r41 - 2013-01-28 - AlexandreBeche